

Karty kryptograficzne w środowisku Linux

Łukasz Piątkowski¹

¹Institut Informatyki
Politechnika Poznańska

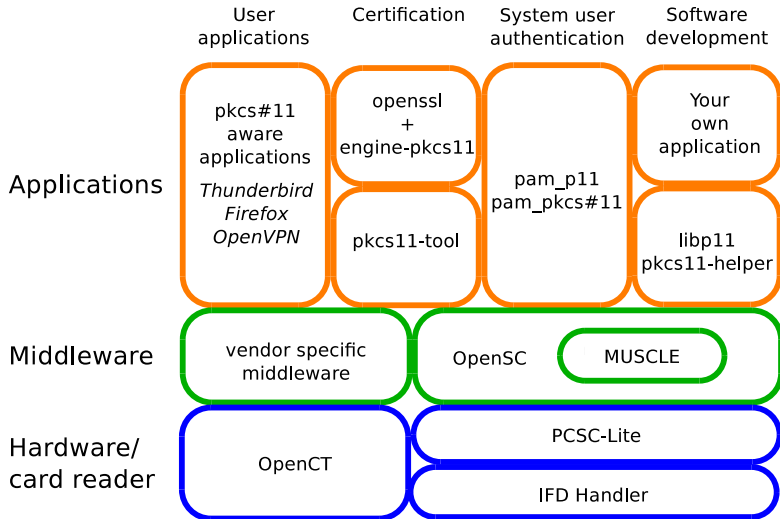
II Krajowa Konferencja Użytkowników Systemów Elektronicznej
Legitymacji Studenckiej, 2010

Plan

- 1 Wprowadzenie
 - Środowisko i narzędzia
 - Czytniki
 - Middleware
 - Aplikacje

- 2 Przykład wykorzystania
 - Certyfikacja

Oprogramowanie, narzędzia



Dostęp do urządzeń

- OpenCT (Open Card Terminal), część projektu OpenSC
 - może działać jako niezależny sterownik sprzętu
 - obsługuje m.in. czytniki USB zgodne z CCID
 - wystawia sterownik dla PC/SC-Lite
 - wspiera CT/API
- PC/SC-Lite
 - wymaga sterownika IFD (ifdhandler)
 - lepiej wspierany przez producentów sprzętu (Omnikey)

Middleware PKCS#11

- OpenSC
 - wsparcie dla niektórych kart Schlumberger, Siemens, Oberthur – bardzo specyficzne wersje
 - wsparcie dla OpenCT
 - wsparcie kart JavaCard przez MUSCLE
- rozwiązania producentkie (osobiście testowane)
 - Oberthur – działa tylko częściowo, znane i nieusuwane błędy, można „zablokować kartę”
 - SafeSign – działa, drobne błędy, sensowny support, wgranie appletu SafeSign wymaga usunięcia innych cert managerów
 - testowane tylko wersje 32 bit

MUSCLE

- całkowicie wolna implementacja obu stron rozwiązania
 - applet na karty zgodne z JavaCard 2.1
 - wgranie appletu przy pomocy programu gpshe
 - PKCS#11 zgodny i działający z OpenSC
 - zastrzeżenia co do jakości kodu appletu
 - ograniczona liczba mechanizmów kryptograficznych

Certyfikacja użytkowników

- możliwa pełna certyfikacja z generowaniem kluczy na karcie
- możliwa integracja z openssl dla generowania żądań i podpisywania certyfikatów
- wymaga dodatkowych narzędzi z projektu OpenSC
 - *pkcs11-tool* – generowanie kluczy, PIN, wgrywanie obiektów i certyfikatów

Logowanie do systemu

- *pam_p11* – prosty moduł PAM uwierzytelniający na podstawie okazanego certyfikatu i certyfikatów wskazanych przez użytkownika
- *pam_pkcs11* – rozbudowany moduł PAM
 - werifikacja CA, CRL
 - mapowanie certyfikatów do loginów
 - mapowania możliwe na podstawie LDAP, Kerberos i wiele innych

Biblioteki i API

- *libp11* – niskopoziomowy interfejs PKCS#11
- *pkcs11-helper* – wrapper dla *libp11*
 - interfejs obsługi czytników i kart
 - interfejs obsługi certyfikatów i obiektów danych

Sprzęt

Czytnik: *Omnikey 3121*
(testowany również *Omnikey 5321*)

- podłączony przez *pcscd* ze sterownikami ze strony producenta

Karta: *Oberthur IDOne Cosmo 64D v.5.4*



Inicjalizacja

- opcja `-module` ładuje bibliotekę realizującą PKCS#11
- w przykładach użyty moduł Oberthur
`/usr/lib/libOcsCryptoki.so`

Listing

```
root@piontec-desktop:~# pkcs11-tool --module  
/usr/lib/libOcsCryptoki.so --init-token --label "testing"  
root@piontec-desktop:~# pkcs11-tool --module  
/usr/lib/libOcsCryptoki.so --init-pin
```

Uwaga

To nie działa z middleware Oberthur

Generowanie pary kluczy

Listing

```
root@piontec-desktop:~# pkcs11-tool --module
/usr/lib/lib0csCryptoki.so -k --key-type rsa:2048 --label
piontec_2 -l
Please enter User PIN:
Key pair generated:
Private Key Object; RSA
label: piontec_k1
ID: 7ad241f677962ffa5024f7dc1b1e654fce1f221e
Usage: decrypt, sign, unwrap
Public Key Object; RSA 2048 bits
label: piontec_k1
ID: 7ad241f677962ffa5024f7dc1b1e654fce1f221e
Usage: encrypt, verify, wrap
```

Konfiguracja openssl

Plik *card.cnf*

```
openssl_conf = openssl_def
HOME = /etc/ssl/card_test
RANDFILE = $ENV::HOME/.rnd
CA_NAME = secpl-card-test
oid_section = new_oids
[openssl_def]
engines = engine_section
[engine_section]
pkcs11 = pkcs11_section
[pkcs11_section]
engine_id = pkcs11
dynamic_path = /usr/lib/engines/engine_pkcs11.so
MODULE_PATH = /usr/lib/lib0csCryptoki.so
```

Prośba o certyfikat i certyfikat

- żądanie certyfikatu

Listing

```
openssl req -config card.cnf -new -key  
id_7ad241f677962ffa5024f7dc1b1e654fcef221e -keyform engine  
-out piontec_k1.req
```

- konwersja między formatami

Listing

```
root@piontec-desktop:/tmp# openssl x509 -inform pem  
-outform der -in piontec_k1.pem -out piontec_k1.der
```

Wgranie certyfikatu na kartę

- wgranie certyfikatu w formacie DER

Listing

```
root@piontec-desktop:~# pkcs11-tool --module
/usr/lib/lib0csCryptoki.so -w piontec_k1.der -y cert
--label piontec_k1 -l
Please enter User PIN:
```

- (opcjonalne) usunięcie samego klucza publicznego

Listing

```
root@piontec-desktop:~# pkcs11-tool --module
/usr/lib/lib0csCryptoki.so --delete-object -y pubkey --id
7ad241f677962ffa5024f7dc1b1e654fce1f221e -l
Please enter User PIN:
```

Weryfikacja

Listing

```
root@piontec-desktop:~# pkcs11-tool --module
/usr/lib/lib0csCryptoki.so -0
Certificate Object, type = X.509 cert
label: piontec_k1
ID: 7ad241f677962ffa5024f7dc1b1e654fce1f221e
```

```
root@piontec-desktop:~# pkcs11-tool --module
/usr/lib/lib0csCryptoki.so -0 -1
Please enter User PIN:
Private Key Object; RSA
label: piontec_k1
ID: a35d11eb40d8eb9d5d9220d6701b90abaf758d9d
Usage: decrypt, sign, unwrap
Certificate Object, type = X.509 cert
label: piontec_k1
ID: a35d11eb40d8eb9d5d9220d6701b90abaf758d9d
```

Podsumowanie

- obsługa kart kryptograficznych w linuxie jest możliwa i jak najbardziej działa
- możliwa realizacja wszystkich podstawowych zadań opartych
 - PKCS#11
- bardzo mało dostępnych materiałów i przykładów
- często kiepskie wsparcie ze strony producentów
- middleware opensource często bardzo specyficzny
- konieczne lepsze wsparcie ze strony producentów

Materiały I

- projekt *OpenSC*: <http://www.opensc-project.org/opensc>
- projekt *OpenCT*: <http://www.opensc-project.org/openct>
- projekt *MUSCLE*: <http://www.linuxnet.com/>
- projekt *pcsc-lite*: <http://pcsclite.alioth.debian.org/>
- reszta projektów ze strony: <http://www.opensc-project.org/>